UNIDAD 1: Práctica 03 - Tipos de objetos: factores, listas y hojas de datos, operadores y funciones que operan sobre ellos.

Abigail Ramos

1. FACTORES NOMINALES Y ORDINALES.

FACTORES NOMINALES.

- Ejemplo 1: Variables sexo (categórica) y edad en una muestra de 7 alumnos del curso.

```
# Supongamos que se obtuvieron los siguientes datos:

sexo <- c("M", "F", "F", "M", "F", "F", "M")
sexo
```

```
## [1] "M" "F" "F" "M" "F" "F" "M"
```

```
edad <- c(19, 20, 19, 22, 20, 21, 19)
edad
```

```
## [1] 19 20 19 22 20 21 19
```

```
# Podemos construir un factor con los niveles o categorias de sexo

FactorSexo = factor(sexo)
FactorSexo
```

```
## [1] M F F M F F M
## Levels: F M
```

```
# Se pueden ver los niveles o categorías del factor con: levels(FactorSexo)
# Crear una tabla que contenga la media muestral por categoría de sexo (nivel del factor):

mediaEdad <- tapply(edad, FactorSexo, mean)
mediaEdad
```

```
##  F  M
## 20 20
```

```
# Note que el primer argumento debe ser un vector, que es del cual se encontrarán las medidas de resume

is.vector(mediaEdad)
```

```
## [1] FALSE
```

```
is.matrix(mediaEdad)
```

```
## [1] FALSE
```

```
is.list(mediaEdad)
```

```
## [1] FALSE
```

```
is.table(mediaEdad)
```

```
## [1] FALSE
```

```
is.array(mediaEdad)
```

```
## [1] TRUE
```

Es de tipo array

FACTORES ORDINALES

Los niveles de los factores se almacenan en orden alfabético, o en el orden en que se especificaron en la función factor() si ello se hizo explícitamente.

A veces existe una ordenación natural en los niveles de un factor, orden que deseamos tener en cuenta en los análisis estadísticos. La función ordered() crea este tipo de factores y su uso es idéntico al de la función factor(). Los factores creados por la función factor() los denominaremos nominales o simplemente factores cuando no haya lugar a confusión, y los creados por la función ordered() los denominaremos ordinales. En la mayoría de los casos la única diferencia entre ambos tipos de factores consiste en que los ordinales se imprimen indicando el orden de los niveles. Sin embargo, los contrastes generados por los dos tipos de factores al ajustar Modelos lineales, son diferentes.

2. CREACIÓN Y MANEJO DE LISTAS.

Una lista es un objeto que contiene una colección ordenada de objetos de diferente tipo (vector, matriz, arreglo, función, o lista), conocidos como componentes. Se construye con la función list(), que tiene la forma general siguiente:

Lista <- list(nombre1 = objeto1, nombre2 = objeto2, . . . , nombren = objeton)

Si omite los nombres, las componentes sólo estarán numeradas. Las componentes pueden accederse por su número o posición, ya que siempre están numeradas, o también pueden referirse por su nombre, si lo tienen.

- Ejemplo 1: Crear una Lista con cuatro componentes.

```
lista1<-list(padre="Pedro", madre="María", no.hijos=3, edad.hijos=c(4,7,9))
lista1
```

```
## $padre
## [1] "Pedro"
##
## $madre
## [1] "María"
##
## $no.hijos
## [1] 3
##
## $edad.hijos
## [1] 4 7 9
```

```
is.matrix(lista1)
```

```
## [1] FALSE
```

```
is.vector(lista1$edad.hijos)
```

```
## [1] TRUE
```

- Ejemplo 2: Acceso a las componentes de una lista:

```
lista1[1] # accede a la componente como una lista (con etiqueta y valor)
```

```
## $padre
## [1] "Pedro"
```

```
lista1["padre"] # el acceso es igual que con lista1[1]
```

```
## $padre
## [1] "Pedro"
```

```
lista1[[2]] # accede al valor o valores de la componente segunda pero no muestra el nombre de la compon
```

```
## [1] "María"
```

```
lista1["madre"] # el acceso es igual que con lista1[[1]]
```

```
## $madre
## [1] "María"
```

- Ejemplo 3: Acceso a los elementos de la cuarta componente:

```
lista1[[4]][2] #(se indica el elemento a ingresar en el segundo corchete)
```

```
## [1] 7
```

- Ejemplo 4: Acceso de las componentes de una lista por su nombre:

```
lista1$padre #similar a
```

```
## [1] "Pedro"
```

```
lista1["padre"]
```

```
## $padre
## [1] "Pedro"
```

Forma general: **Nombre_de_lista$nombre_de_componente** Por ejemplo:

```
lista1$padre #equivale a
```

```
## [1] "Pedro"
```

```
lista1[[1]]
```

```
## [1] "Pedro"
```

```
lista1$edad.hijos[2] #equivale a
```

```
## [1] 7
```

```
lista1[[4]][2]
```

```
## [1] 7
```

- Ejemplo 5: Utilizar el nombre de la componente como índice:

```
lista1[["nombre"]] #se puede ver que equivale a
```

```
## NULL
```

```
lista1$nombre
```

```
## NULL
```

```
#También es útil la forma:
x <- "nombre"; lista1[x]
```

```
## $<NA>
## NULL
```

- Ejemplo 6: Creación de una sublista de una lista existente:

```
subLista <- lista1[4]
subLista
```

```
## $edad.hijos
## [1] 4 7 9
```

- Ejemplo 7: Ampliación de una lista: por ejemplo, la lista lista1 tiene 4 componentes y se le puede agregár una quinta componente con:

```
lista1[5] <- list(sexo.hijos=c("F", "M", "F"))
lista1
```

```
## $padre
## [1] "Pedro"
##
## $madre
## [1] "María"
##
## $no.hijos
## [1] 3
##
## $edad.hijos
## [1] 4 7 9
##
## [[5]]
## [1] "F" "M" "F"
```

Observe que no aparece el nombre del objeto agregado, pero usted puede modificar la estructura de la lista lista1 con:

```
lista1 <- edit(lista1)
```

Nota: Se puede aplicar la función data.entry() para modificar la estructura de una lista.

- Ejemplo 8: Funciones que devuelven una lista. Las funciones y expresiones de R devuelven un objeto como resultado, por tanto, si deben devolver varios objetos, previsiblemente de diferentes tipos, la forma usual es una lista con nombres. Por ejemplo, la función eigen() que calcula los autovalores y autovectores de una matriz simétrica.

Ejecute las siguientes instrucciones:

```
S <- matrix(c(3, -sqrt(2), -sqrt(2), 2), nrow=2, ncol=2)
S
```

```
##           [,1]      [,2]
## [1,]  3.000000 -1.414214
## [2,] -1.414214  2.000000
```

```r
autovS <- eigen(S)
autovS
```

```
## eigen() decomposition
## $values
## [1] 4 1
##
## $vectors
##            [,1]       [,2]
## [1,] -0.8164966 -0.5773503
## [2,]  0.5773503 -0.8164966
```

Si quisiéramos almacenar sólo los autovalores de S, podemos hacer lo siguiente:

```r
evals <- eigen(S)$values
evals
```

```
## [1] 4 1
```

- Ejemplo 9: Crear una matriz dando nombres a las filas y columnas

```r
Notas <- matrix(c(2, 5, 7, 6, 8, 2, 4, 9, 10), ncol=3, dimnames=list(c("Matemática","Álgebra","Geometría
c("Juan","José","René")))
Notas
```

```
##            Juan José René
## Matemática    2    6    4
## Álgebra       5    8    9
## Geometría     7    2   10
```

```r
# Los nombres se dan primero para filas y luego para columnas.
```

3. CREACIÓN Y MANEJO DE HOJAS DE DATOS (DATA FRAME).

- Ejemplo 1: Creación de un data frame teniendo como columnas tres vectores: **En primer lugar generamos los tres vectores** El primer vector tendrá 20 elementos que se obtienen con reemplazamiento de una muestra aleatoria de valores lógicos.

```r
log <- sample(c(TRUE, FALSE), size = 20, replace = T)
log
```

```
##  [1]  TRUE  TRUE FALSE FALSE  TRUE FALSE  TRUE FALSE FALSE  TRUE  TRUE FALSE
## [13]  TRUE FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
```

```r
# Note que puede usar T en lugar de TRUE y F en lugar de FALSE.
```

El segundo vector tendrá 20 elementos de valores complejos cuya parte real proviene de una distribución Normal estándar y cuya parte imaginaria lo hace de una distribución Uniforme(0,1)

```
comp <- rnorm(20) + runif(20) * (1i)
comp
```

```
##  [1]  1.5123270+0.3874013i -0.2174571+0.0988639i  0.7193462+0.7243080i
##  [4]  0.4909967+0.0991026i -0.9339657+0.2436650i -1.3333753+0.6345504i
##  [7] -0.1915875+0.3482644i -1.1730715+0.4658045i -1.8491521+0.3187243i
## [10]  1.1828321+0.8364248i  0.2581484+0.7645244i  0.0547040+0.4930372i
## [13] -1.8854133+0.1810816i -1.5739192+0.3104606i -1.2068336+0.4135618i
## [16]  1.3491420+0.5122842i  0.3276063+0.1208107i -0.8203343+0.6666645i
## [19]  0.0999480+0.2246525i  0.7685259+0.0515812i
```

El tercer vector tendrá 20 elementos de una distribución Normal estándar

```
num <- rnorm(20, mean=0, sd=1)
num
```

```
##  [1]  0.3206186  0.1098500 -0.5193929 -1.4588835  2.3782950 -0.1809827
##  [7]  0.2616328  0.5090555 -1.3951744 -0.1024712 -0.1048075 -0.5557651
## [13]  1.0360078 -1.4779716  1.0688305 -0.5366317  0.7658856  0.4821782
## [19] -0.6768839  0.6014830
```

Crear un data frame compuesto por los tres vectores anteriores

```
df1 <- data.frame(log, comp, num)
df1
```

```
##      log                  comp        num
## 1   TRUE  1.5123270+0.3874013i  0.3206186
## 2   TRUE -0.2174571+0.0988639i  0.1098500
## 3  FALSE  0.7193462+0.7243080i -0.5193929
## 4  FALSE  0.4909967+0.0991026i -1.4588835
## 5   TRUE -0.9339657+0.2436650i  2.3782950
## 6  FALSE -1.3333753+0.6345504i -0.1809827
## 7   TRUE -0.1915875+0.3482644i  0.2616328
## 8  FALSE -1.1730715+0.4658045i  0.5090555
## 9  FALSE -1.8491521+0.3187243i -1.3951744
## 10  TRUE  1.1828321+0.8364248i -0.1024712
## 11  TRUE  0.2581484+0.7645244i -0.1048075
## 12 FALSE  0.0547040+0.4930372i -0.5557651
## 13  TRUE -1.8854133+0.1810816i  1.0360078
## 14 FALSE -1.5739192+0.3104606i -1.4779716
## 15  TRUE -1.2068336+0.4135618i  1.0688305
## 16  TRUE  1.3491420+0.5122842i -0.5366317
## 17  TRUE  0.3276063+0.1208107i  0.7658856
## 18 FALSE -0.8203343+0.6666645i  0.4821782
## 19 FALSE  0.0999480+0.2246525i -0.6768839
## 20 FALSE  0.7685259+0.0515812i  0.6014830
```

Crear un vector de nombres de los tres vectores anteriores

```
nombres <- c("logico", "complejo", "numerico")
```

Define los nombres de las columnas del data frame asignándoles el vector nombres

```
names(df1) <- nombres
df1
```

```
##    logico              complejo   numerico
## 1    TRUE  1.5123270+0.3874013i  0.3206186
## 2    TRUE -0.2174571+0.0988639i  0.1098500
## 3   FALSE  0.7193462+0.7243080i -0.5193929
## 4   FALSE  0.4909967+0.0991026i -1.4588835
## 5    TRUE -0.9339657+0.2436650i  2.3782950
## 6   FALSE -1.3333753+0.6345504i -0.1809827
## 7    TRUE -0.1915875+0.3482644i  0.2616328
## 8   FALSE -1.1730715+0.4658045i  0.5090555
## 9   FALSE -1.8491521+0.3187243i -1.3951744
## 10   TRUE  1.1828321+0.8364248i -0.1024712
## 11   TRUE  0.2581484+0.7645244i -0.1048075
## 12  FALSE  0.0547040+0.4930372i -0.5557651
## 13   TRUE -1.8854133+0.1810816i  1.0360078
## 14  FALSE -1.5739192+0.3104606i -1.4779716
## 15   TRUE -1.2068336+0.4135618i  1.0688305
## 16   TRUE  1.3491420+0.5122842i -0.5366317
## 17   TRUE  0.3276063+0.1208107i  0.7658856
## 18  FALSE -0.8203343+0.6666645i  0.4821782
## 19  FALSE  0.0999480+0.2246525i -0.6768839
## 20  FALSE  0.7685259+0.0515812i  0.6014830
```

Define los nombres de las filas del data frame asignándoles un vector de 20 elementos correspondientes a las 20 primeras letras del abecedario

```
row.names(df1) <- letters[1:20]
df1
```

```
##   logico              complejo   numerico
## a   TRUE  1.5123270+0.3874013i  0.3206186
## b   TRUE -0.2174571+0.0988639i  0.1098500
## c  FALSE  0.7193462+0.7243080i -0.5193929
## d  FALSE  0.4909967+0.0991026i -1.4588835
## e   TRUE -0.9339657+0.2436650i  2.3782950
## f  FALSE -1.3333753+0.6345504i -0.1809827
## g   TRUE -0.1915875+0.3482644i  0.2616328
## h  FALSE -1.1730715+0.4658045i  0.5090555
## i  FALSE -1.8491521+0.3187243i -1.3951744
## j   TRUE  1.1828321+0.8364248i -0.1024712
## k   TRUE  0.2581484+0.7645244i -0.1048075
## l  FALSE  0.0547040+0.4930372i -0.5557651
## m   TRUE -1.8854133+0.1810816i  1.0360078
## n  FALSE -1.5739192+0.3104606i -1.4779716
## o   TRUE -1.2068336+0.4135618i  1.0688305
## p   TRUE  1.3491420+0.5122842i -0.5366317
```

```
## q   TRUE  0.3276063+0.1208107i  0.7658856
## r  FALSE -0.8203343+0.6666645i  0.4821782
## s  FALSE  0.0999480+0.2246525i -0.6768839
## t  FALSE  0.7685259+0.0515812i  0.6014830
```

- Ejemplo 2: Vamos a crear la siguiente hoja de datos que tiene 4 variables o columnas:

```
edad <- c(18, 21, 45, 54)
edad
```

```
## [1] 18 21 45 54
```

```
datos <- matrix(c(150, 160, 180, 205, 65, 68, 65, 69), ncol=2, dimnames=list(c(), c("Estatura","Peso")))
datos
```

```
##      Estatura Peso
## [1,]      150   65
## [2,]      160   68
## [3,]      180   65
## [4,]      205   69
```

```
sexo <- c("F", "M", "M", "M")
sexo
```

```
## [1] "F" "M" "M" "M"
```

```
hoja1 <- data.frame(c(Edad=edad, datos, Sexo=sexo))
hoja1
```

```
##    c.Edad...edad..datos..Sexo...sexo.
## 1                                  18
## 2                                  21
## 3                                  45
## 4                                  54
## 5                                 150
## 6                                 160
## 7                                 180
## 8                                 205
## 9                                  65
## 10                                 68
## 11                                 65
## 12                                 69
## 13                                  F
## 14                                  M
## 15                                  M
## 16                                  M
```

```
#Para editar o agregar datos, o componentes utilice: fix(hoja1)
```

Nota: Puede forzar que una lista, cuyos componentes cumplan las restricciones para ser una hoja de datos, realmente lo sea, mediante la función as.data.frame()

Conexión de listas o hojas de datos.

La función search() busca y presenta qué hojas de datos, listas o bibliotecas han sido conectadas o desconectadas. Teclee search()

```
search()
```

```
## [1] ".GlobalEnv"        "package:stats"     "package:graphics"
## [4] "package:grDevices" "package:utils"     "package:datasets"
## [7] "package:methods"   "Autoloads"         "package:base"
```

La función attach() es la función que permite conectar en la trayectoria de búsqueda no sólo directorios, listas y hojas de datos, sino también otros tipos de objetos. Teclee attach(hoja1) y luego search()

```
attach(hoja1)
search()
```

```
##  [1] ".GlobalEnv"        "hoja1"             "package:stats"
##  [4] "package:graphics"  "package:grDevices" "package:utils"
##  [7] "package:datasets"  "package:methods"   "Autoloads"
## [10] "package:base"
```

Luego puede acceder a las componentes por su nombre:

```
#hoja1$Peso <- Peso+1; hoja1
```

Posteriormente podrá desconectar el objeto utilizando la función detach(), utilizando como argumento el número de posición o, preferiblemente, su nombre. Teclee detach(hoja1) y compruebe que la hoja de datos ha sido eliminada de la trayectoria de búsqueda con search().